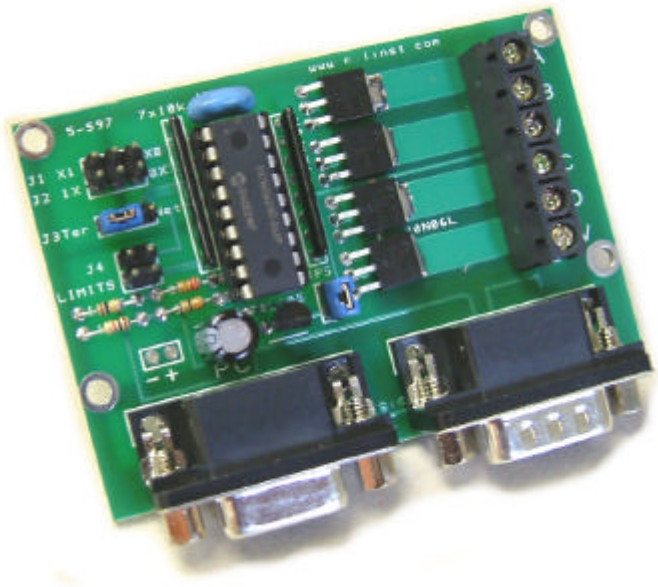
MILFORD INSTRUMENTS Ltd

Uni-Polar Stepper Motor Driver, Part Number 5-597



The Stepper Motor driver provides a convenient way of controlling standard Uni-Polar stepper motors rated at up to 5A and 24V DC in one of two modes:
In Terminal Mode the motor may be simply controlled from a standard terminal programme such as Terminal (Win 3.1) or Hyper-terminal (Win '9X).
In Network Mode, several motor driver boards may be connected onto a single RS232 control line for multiple motor applications.
The driver board only works in full step mode and does not support half-stepping.

Set-up

Communications

The Driver board expects RS232 type signals at 9600 baud, 8-bits, no parity and 1 stop bit. There is no handshaking (either hardware or software). When using Terminal or Hyper-Terminal, please ensure the system settings reflect the above parameters otherwise it will not be possible to control the motors.
Connection to a PC should be by via a standard Modem type 9-way cable (NOT a NULL Cable).
Multiple board may be linked together by additional 9-way cable sets.

Stepper Motor

The driver card will drive unipolar stepper motors with ratings up to 5A and 24V
Connect the stepper motor to the A, B, C and D connection points. A - B are phase windings one and C - D the second phase windings. Connect the centre taps of both windings to the +V contact.

Reg Office: Milford House, 120 High Street, South Milford, LEEDS LS25 5AQ United Kingdom
Reg Number: 4222329 tel: 01977 683665 fax: 01977 681465 www.milinst.com

Power Supply
Connect a suitable supply for the stepper motor to the +V and –V connections.
For supplies voltages in the range of 7.5 to 15V DC, the driver electronics may be run from the same power supply by inserting the jumper onto the J5 contacts. For supply voltages outside this range, it will be necessary to power the electronics via a separate 9V supply connected to the -/+ pads at the bottom left of the board.
You must remove the J5 jumper if powering the electronics separately.

Jumper settings

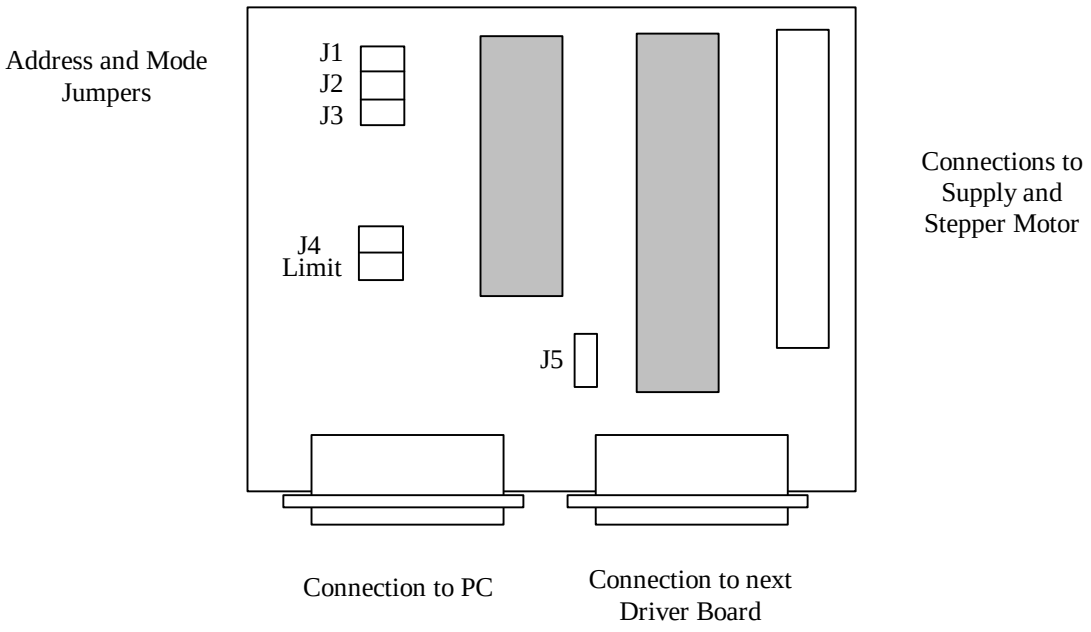
Mode Select
Network mode is selected with jumper 3 set to the NET position
Terminal mode is selected with jumper 3 set to the TER position

Driver Board Address (only applicable in Network mode)
Driver Board address is determined by jumpers 1 and 2:

Board Address	Jumper 2	Jumper 1
00 (0)	IN	IN
01 (1)	IN	OUT
10 (2)	OUT	IN
11 (3)	OUT	OUT

Note that the address jumpers are only read at power-up. Changes to the address after power-up will not be recognised.

Limit Switch
Limit switches may be used to protect the driver board driving the motor out of bounds- the driver board checks the status of the limit switch pin after every step. If the pin is pulled low then following moves are aborted.
Limit switches must be normally open and connected in parallel across the LIMIT header.
In Network mode only, closed limit switches may be ignored to allow for datum placing at set-up etc. Please refer to the Network Commands section for further details.



Command Set

Terminal Mode
Ensure the Mode jumper (J3) is set to the TER position
Set up your PC's Terminal (or Hyper-Terminal) programme to 9600 baud, no handshaking and 8-bits,no parity and 1 stop bit. Turn on "local echo" so the PC displays your commands as you type them
Commands to the driver board are in the following format:

+500 20<return>

Rotate the motor forward 500 steps with a 20 milliseconds gap between steps.
To run the stepper backward, substitute - for +.
To free-wheel the motor, send *XX XX where XX are any digits (ignored by the driver board). The motor coils are automatically re-energised when the next +/- command is sent.

The number of steps may be up to 65536 (no commas) and the time gap between steps from 2 to 255 milliseconds. Motor torque depends on the time interval between steps- please refer to the section on Stepper Motors.

Following completion of a command, the driver board returns the LF,CR,"S" prompt. Commands typed in whilst the driver board is stepping the motor are ignored.

If end of travel limit switches are connected, activation of these will cause the immediate response of "L,steps" where steps is the number of steps moved before the activation occurred.

Network Mode
Ensure the Mode jumper is IN
Set the communications protocol to 9600 baud, 8-bits, no parity, one stop bit and no handshaking.

The command structure in Network mode is as follows:

A 4-byte string is sent:
Byte 1:-
bits 6/7= card address (0 to 3)
bits3/5= command code
- 011 forwards with limits active
- 010 backwards with limits active
- 001 disable motor
- 000 report status
- 111 forwards ignoring limits
- 110 backwards ignoring limits
- 101 not defined
- 100 not defined

bits 0/2= step delay period:
- 000 1msec
- 001 2msec
- 010 4msec
- 011 8msec
- 100 16msec
- 101 32msec
- 110 64msec
- 111 128msec

Byte2- high byte of steps
Byte3- low byte of steps
Byte4- crc error sum = b1^b2^b3 ie it is the bitwise XOR performed on a byte by byte basis:

eg B1=%10000001, B2=%00001000, B3=%00000100

B1	%10000001	
B2	%00001000	
B1^B2	%10001001	
B3	%00000100	
B1^B2^B3	%10001101	=B4

Once a 4-byte command set has been received, the driver board checks the address and crc bytes. If the crc byte is OK and the address matches its physical address,, it acknowledges the command with the response “A,addr” where addr is the address number (0-3). If no response is received the “Report Status” command should be sent. If a CRC error was encountered a response of “C X X” will be returned where X is any byte and is to be ignored.

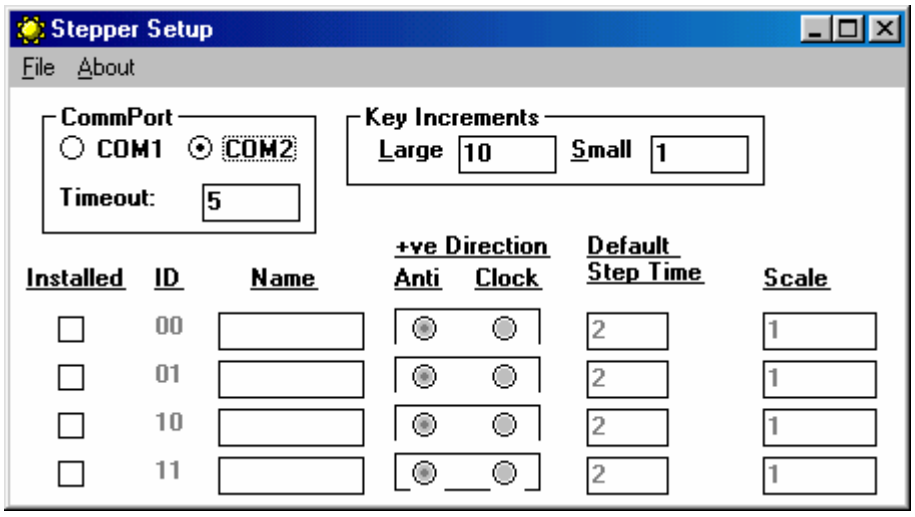
Once a valid command is received, the driver will commence stepping the motor until the command is complete or (if the limit switches are active) a limit switch is activated.

Commands sent whilst the driver is busy will be ignored. The board should be periodically polled by sending the “report status” command to ascertain when a particular command is complete.
If the command was completed satisfactorily, then the “report status” command will return “A,addr” and “R X X” where X is an unspecified byte and is ignored.
If an activated limit switch halted operation, the response will be “A.addr” followed by “L X X “ where X is an unspecified byte and is ignored.
A “report status” command with the motor de-energised will report “A,addr” followed by “F X X “ where X is an unspecified bytes and to be ignored.

Stepper Software

Software is available for controlling the stepper motor drivers, from the Milford Instruments website. (www.milinst.com\download.htm)

On starting the application you will be presented with the following screen



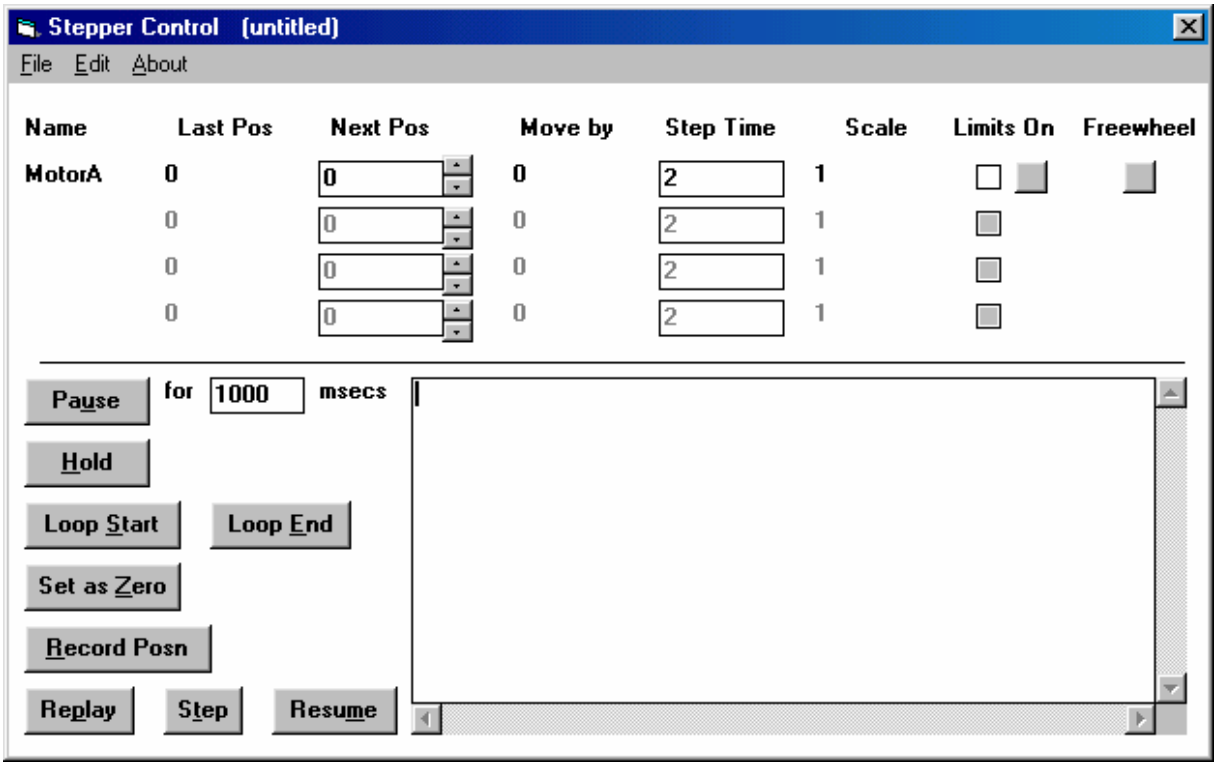
This screen is used to allocate names for the motor drivers that you are using and allocate general characteristics to your stepper motor network.

Indicate the COM port to which your network is connected, and set the timeout period (in seconds). The timeout period is the length of time the program will wait to receive an acknowledgement signal from the network before raising an error message. The default is 5 seconds.

Next, click on the checkboxes relevant to the connected driver cards, and give them names. The names cannot contain spaces or be the same as scripting commands. The names are used to identify motors in the scripting language.
You can also set which direction the motors will go when sent in the 'positive' direction. You can set the default stepping time and you can set a scale factor. This is useful if you have connected the stepper motor to a mechanism such that a rotation of a number of steps is directly proportional to a number of units of movement of another kind. By using the scale factor you can save a lot of time on converting from one type to another.
E.G. If 4 steps of a motor corresponds to 1mm of real movement, then set your scale to 4, and you can enter all numbers in the script as if they were millimetres.

The 'Key Increments' box applies to the distance a motor rotates when you control it with the computer keyboard.

Once the network is set-up, click on Run in the File menu to write and run scripts.



You are now ready to create, edit, save, load and run scripts to control you stepper motor network. Script commands can be entered manually in the large textbox, or by using the tools surrounding it.

Scripting Commands

Command	Function
LIMITON *motor*	Makes the driver board for *motor* respond to limit switches when moving
LIMITOFF *motor*	Allows the driver board for *motor* to ignore limit switches when moving
SETZERO	Sets the datum level for subsequent movements to the current position
PAUSE *val*	Waits for a period of *val* milliseconds
HOLD	Pauses the running of the script indefinitely until the user responds

OFF *motor*	Allows *motor* to freewheel until it is next commanded to move
Motor val1,val2	Makes *motor* move to position *val1* with a step time *val2*
LOOPSTART	Marks the start of a loop
LOOPEND	Marks the end of the loop. Returns to LOOPSTART and begins again

A few notes:
Only one Loop is allowed in a script. When a LOOPEND is reached, the script goes back to the first LOOPSTART it finds.
If you wish to reenergize a motor without moving it, you can simply send it a command to go to its current position and this will do the trick.
Writing commands directly into the script does not affect any of the steppers until the script is run (unlike using the tools). Hence, if you write a LIMITON command, and then use tools to move the stepper, it may ignore the limit switches (unless the relevant checkbox is ticked.)

Tools
There are a number of tools that make writing scripts a little easier.
Various buttons will insert commands automatically. By default, new commands are added to the end of the script. However, if the insertion point is not at the end, i.e. you have stepped through the script or clicked on a line in the script, then the command will be inserted on the next line.
There are a number of methods of setting the position of a stepper motor. You can write the position in the relevant box. Pressing return (or clicking out of the box) will send the command to the driver. You can use the arrows at the side of the box to increment by the small value set on the set-up page. You can use the keyboard:

MOTOR	**00**	**01**	**10**	**11**
UP	Q	W	E	R
DOWN	A	S	D	F

When Shift is held down, the increment is large, if not, the increment is small.
These commands will make the motor move, but the script will not be altered until the Record Position button is pressed. When this is done, the position of all installed stepper motors will be added to the script (even the ones that haven't moved.)

If you make many alterations with the tools in quick succession, you may get a Timeout error. This is because the program is trying to send a lot of information to the networked cards and can be avoided by ensuring that each motor has responded to a tools action before attempting another alteration.

Running Scripts
Once a script is written, you can run it all the way through from the beginning by pressing 'Replay'. If you only wish to execute one command at a time, press 'Step'. To stop a running script, press 'Abort'. To restart a script from the current position press 'Resume'.

You can save and load scripts by using the relevant commands found in the File menu. Scripts are saved as text files that can be looked at using any ASCII text editor.

Tips
The scripting language is designed to send commands to the motor drivers and ensure that the command is received and understood. It does not check to see if the command is completed before going on to the next command in the script. So, if you are sending a series of commands to a single motor, you run the risk of generating a timeout error. This occurs if the execution of the first command takes longer to complete than the timeout period you have set. The scripting program will sit waiting for the motor driver to report that it is ready for the next command only until the timeout criteria is met.

To avoid timing out, you can either set the timeout period to be longer, or insert a PAUSE command of a reasonable length, so that when the program asks if the motor driver is ready, it doesn't have to wait so long.
Another upshot of this is that if consecutive commands are for different drivers, the second command may be sent before the first driver finishes. If this is a problem, there are two methods of getting round it. First, you can insert an appropriate PAUSE command. Better still (assuming you don't create a timeout) is putting a command to the first motor driver to move to its final position (i.e. don't move), between the commands. The scripting programme then waits until the first motor has finished before telling it not to move and going on to the command for the second motor driver.

An example:

Motor A 100,25
Motor B 100,25
Motor C 100,25
Motor A 0,15

Command 1 will be executed. Command 2 will then be executed without waiting for the first driver to send its motor all the way to position 100. Command 3 will follow in the same manner. Command 4, however, will not be executed until the first driver has sent its motor to position 100 and reported that it is ready for the next command. The second and third drivers may still be in the process of sending their motors to position 100

Another example:

Motor A 100,25
Motor B 100,25

Read above to find out what this does. What if you wanted the first motor to stop before the second motor starts? If the timeout setting is high enough, you could do this:

Motor A 100,25
Motor A 100,2
Motor B 100,25

The first command is executed, but the second command cannot be executed until the first driver reports that it is ready. Hence, the third command will not be executed until the first motor has stopped.

Timeouts
There are two types of timeout. Timeout 001 is generated when the program sends a request to a motor driver asking if it is ready to receive a command and times out before it gets a reply.
Timeout 002 is when the program is told that the driver is ready but does not receive and acknowledgement that the command was understood.